Performance Evaluation

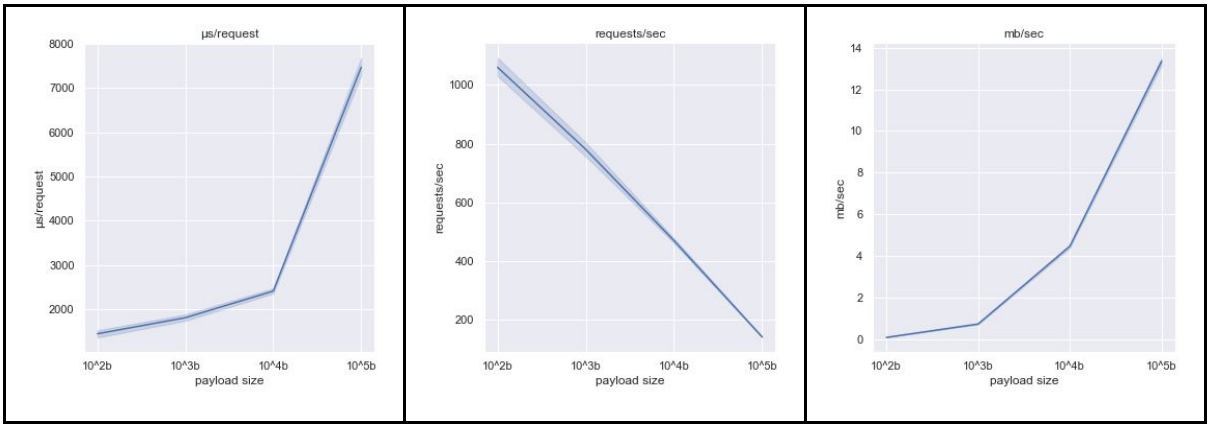
Cloud Databases - Group 4 - Milestone 3

General Remarks

The following performance experiments have been conducted on a cluster with 5 nodes for servers and 3 nodes for clients. The nodes are n1-standard-1 Google Cloud nodes with one 2.6Ghz Xeon Core, 4GB RAM, 1000MBit network and non-SSD disks. Shaded areas around line plots demark 95% confidence intervals of the measurements.

Dependence of Write Performance on Payload Size

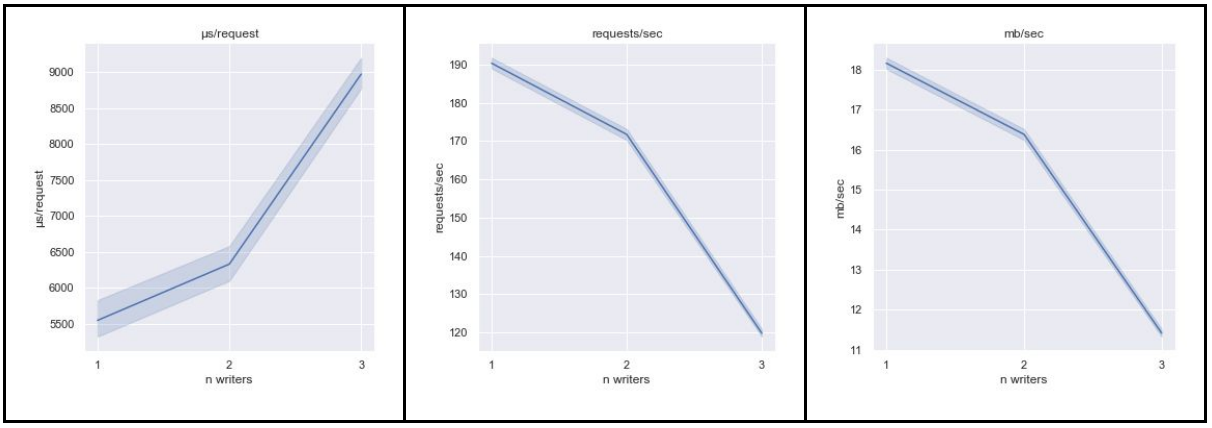
Experimental Setup: We have 1 server and 1 client to produce a write load. We change the size of the payload and analyze how this impacts the performance.



Result: The request latency increases with increasing payload size expectedly. We see that the overall data throughput however increases significantly with increasing payload size. The main reason for this is most likely the rather primitive disk storage mechanism which writes each key-value-pair in a separate file and is globally synchronized. Ideas for improvement would be localizing synchronization to smaller key ranges to allow more concurrency and using a faster append-only write-ahead-log (WAL) for disk persistence.

Degradation of Write Performance under Increasing Load

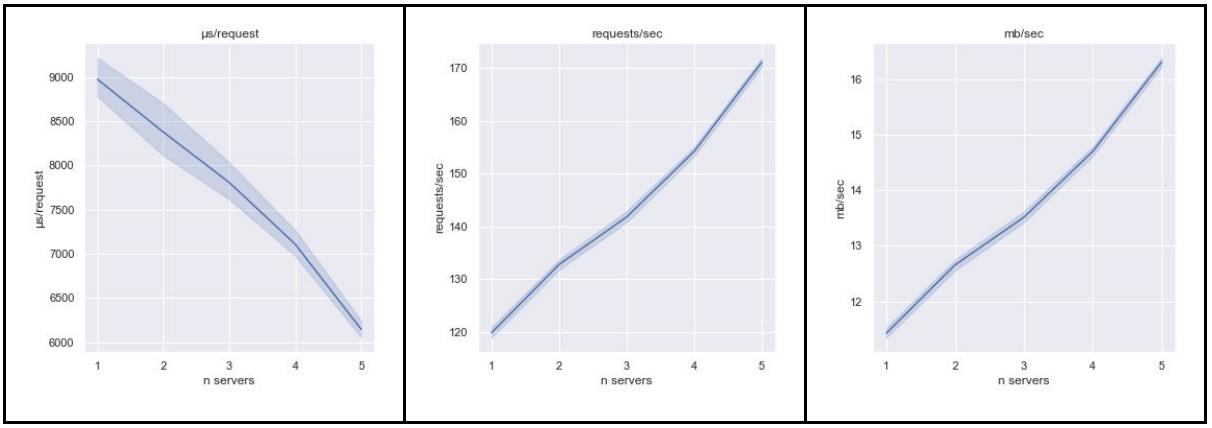
Experimental Setup: We have 1 server and increasingly add clients that send messages with 10^6 bytes payload as fast as possible. We measure how the performance changes on client side.



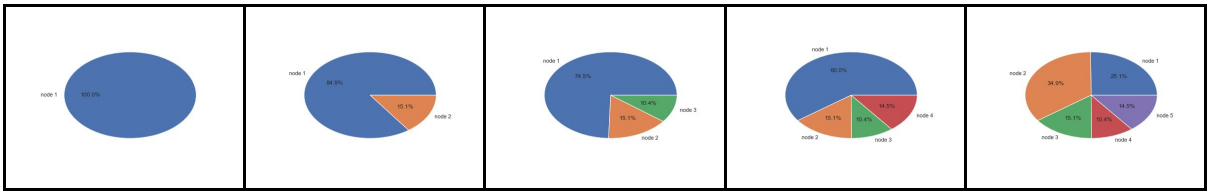
Result: It can be seen how an increasing write load degrades the performance globally.

Improvement of Write Performance by Additional Servers

Experimental Setup: We continue with the final version of the previous experiment where we had 1 server and 3 writer threads. We increasingly add additional server nodes and analyze how the performance changes.



Result: The write performance increases almost linearly with added servers in our case. It is also interesting to look at how the responsibilities for the key spectrum evolve for our cluster under an increasing number of nodes:

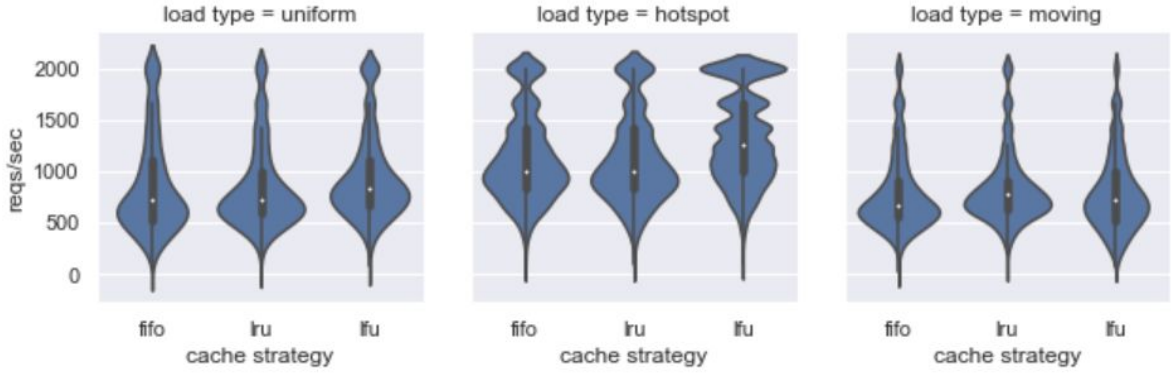


We can see that for small number of nodes, it is not at all safe to assume that they would be responsible for equal sizes of the key spectrum. Only with 5 nodes the parts begin to converge in our case.

Caching Strategies and Reading Performance

Experimental Setup: We have one server node and one reading client. The client accesses keys with one of 3 strategies:

- uniform: Keys are selected randomly from the whole key range.
- hotspot: The client accesses a key from the first percent of the key range with 70% probability. These could represent celebrity profiles on Facebook for example.
- moving: Every second we select a most recent item and only select the one percent of the key range that directly precedes this element. This could e.g. represent a workload for a news company where people usually access only recent articles.



Result: We see that the least frequently used (LFU) cache replacement strategy works well for the hotspot-type workload while the least recently used (LRU) strategy has a certain advantage for the moving-type workload. LFU also does pretty well for the uniform-type workload which we can not directly explain.